

PRAQMA

Testing Microservices

Techniques for Automation

European Testing Conference 2018

Emily Bache

@emilybache

Emily Bache

Practice Lead for Test Automation

Consultant

Author of

“The Coding Dojo Handbook”

@emilybache

emily.bache@pragma.com



Pagero Online
Cloud Service



Invoice
Issuer



Invoice
Recipient



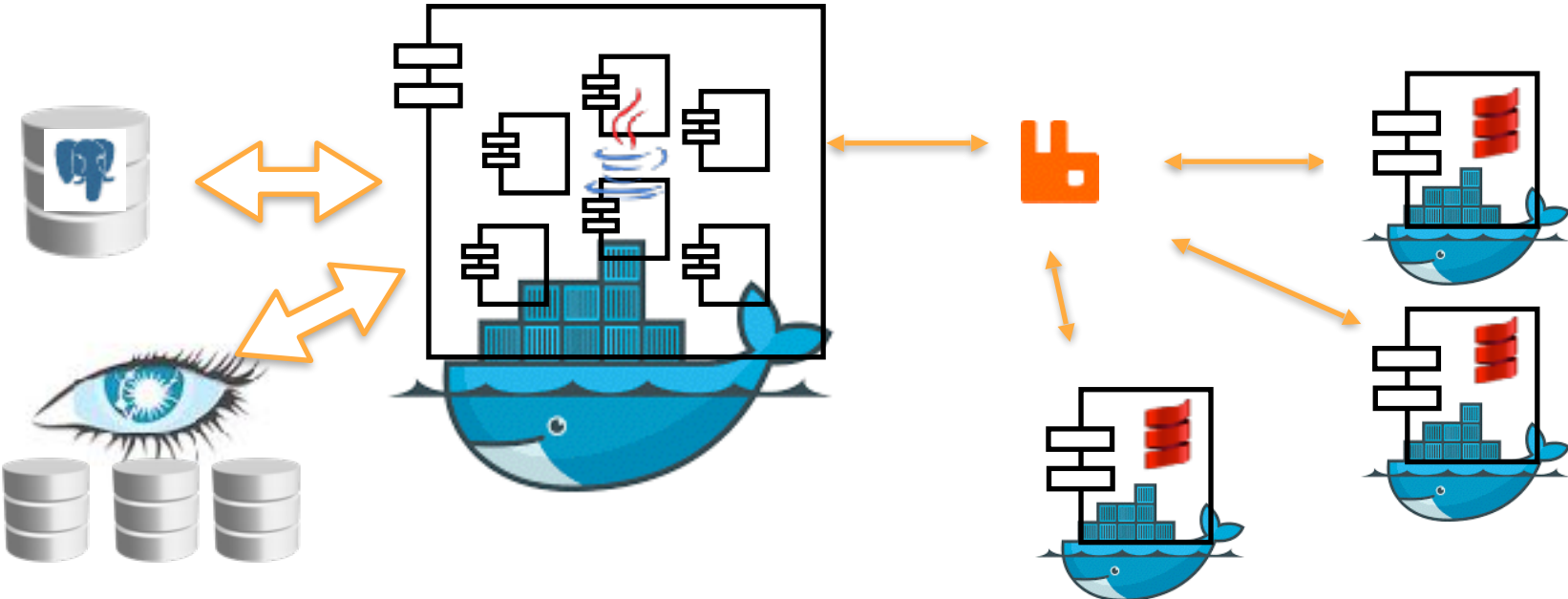
Some numbers...

- We were around 45 developers, in 7 teams
- We had around 60 microservices
- Our monolith was in production for nearly 10 years before we started with microservices, 3-4 years ago

Transaction Growth

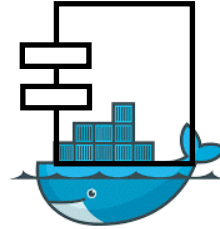


Pagero Online's architecture



A Microservice

- Provides one business capability
- Independently Deployable
- Lightweight API
- Not too big: fits in my head

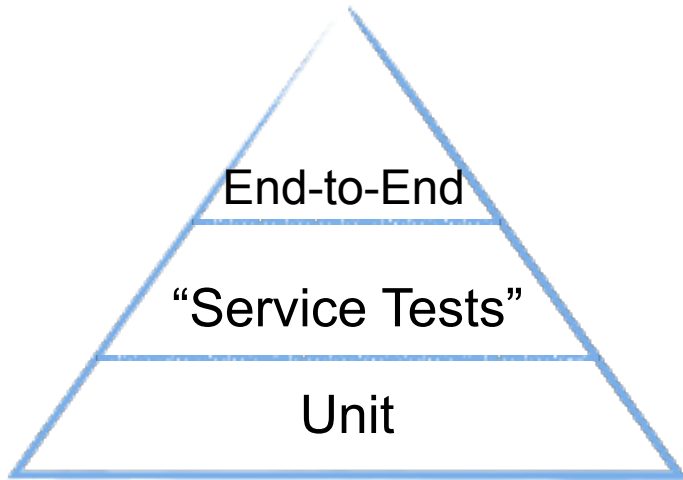


Docker is used to containerise the service

Testing a Microservices Architecture

In my experience

Test Pyramid: Mike Cohn



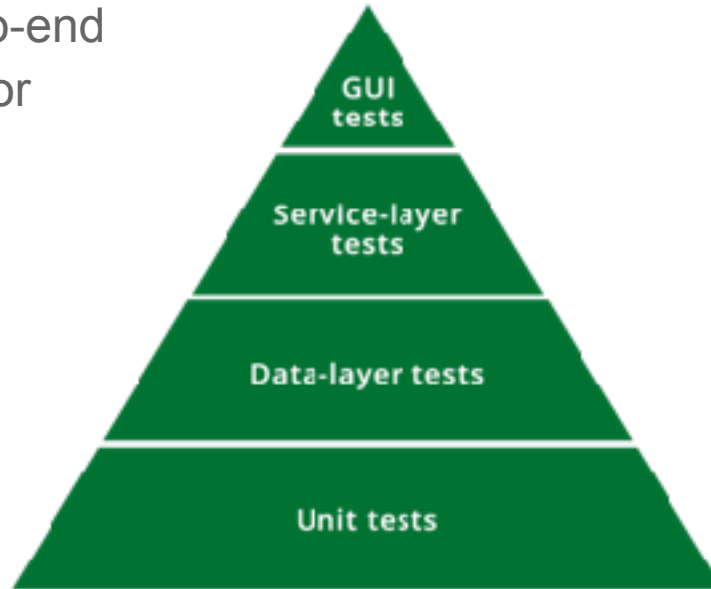
End-To-End tests exercise the whole system

Unit tests are for individual functions

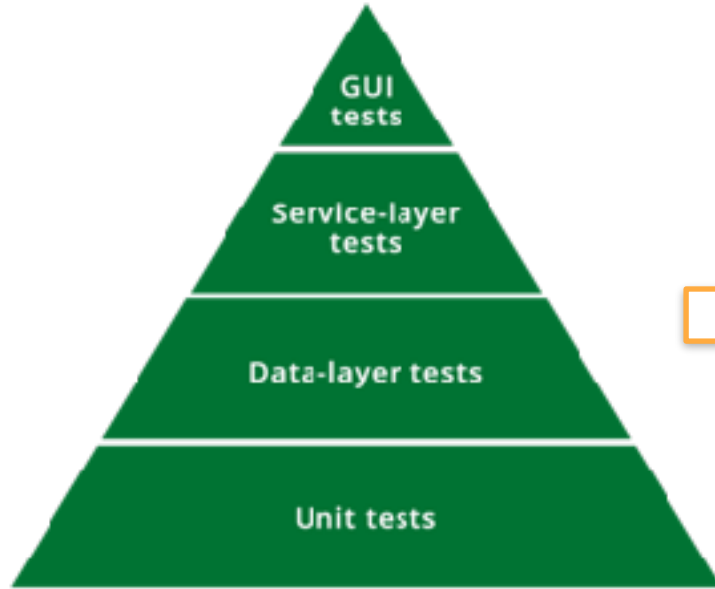
<https://www.mountaingoatsoftware.com/blog/the-forgotten-layer-of-the-test-automation-pyramid>

Testing Strategy for the Monolith

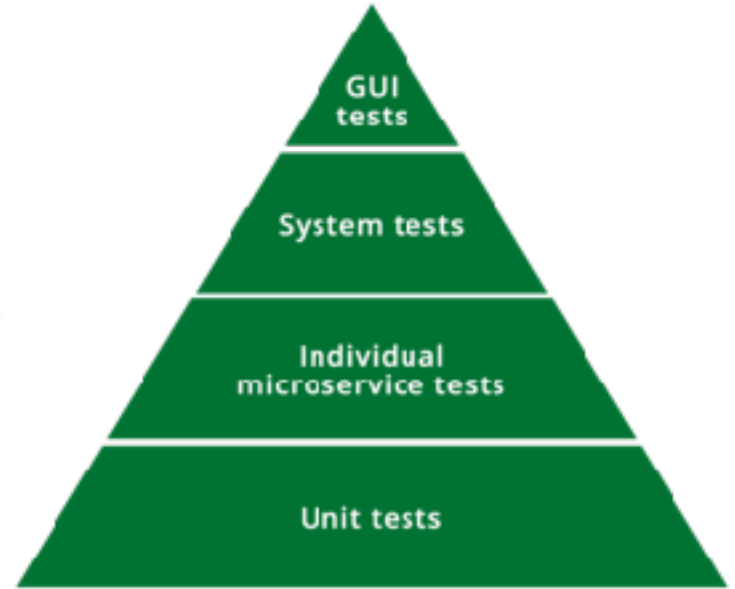
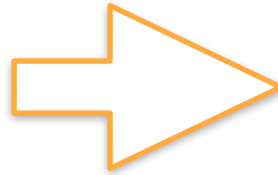
Unit tests and end-to-end
GUI tests still valid for
Microservices
architecture



Testing Strategy

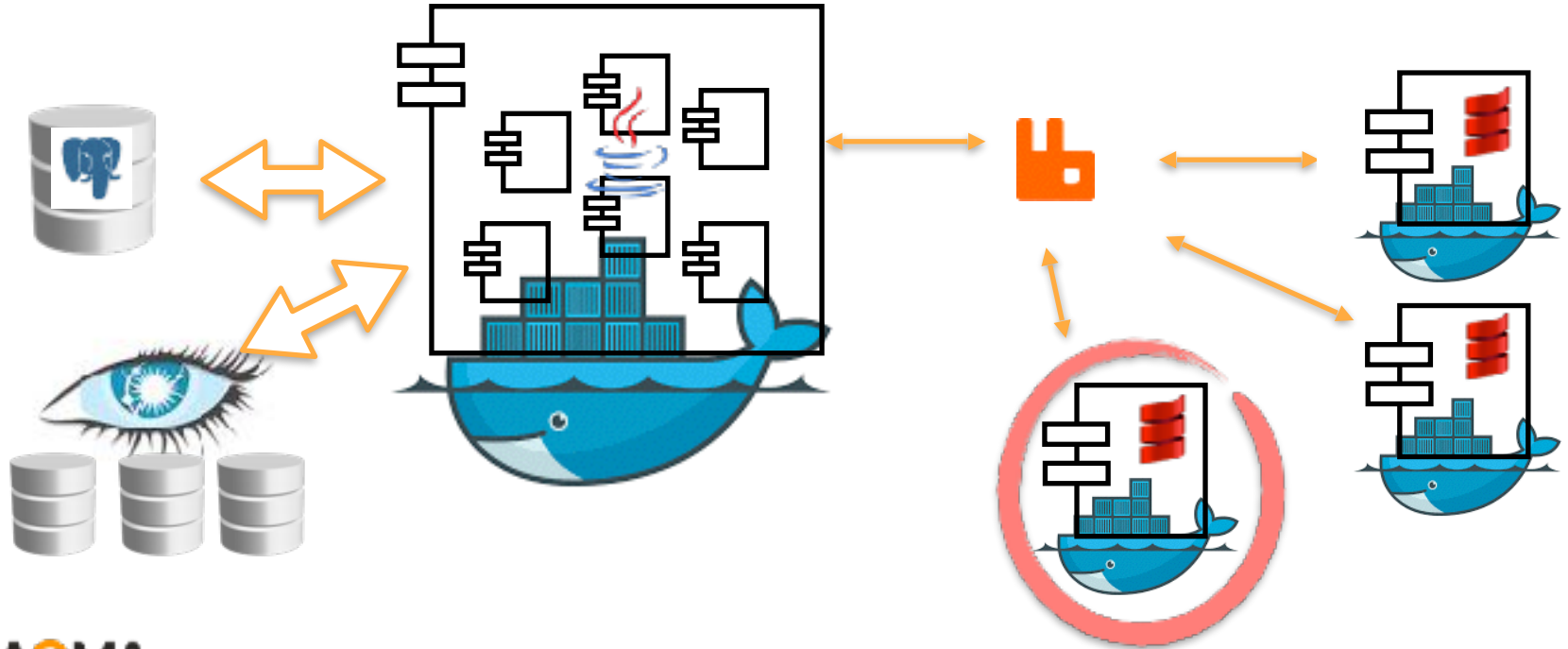


Testing a monolith



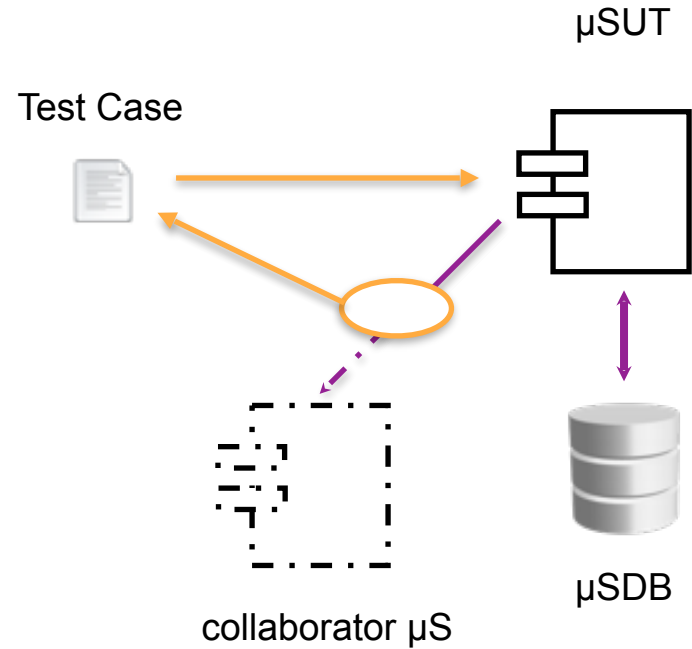
Testing a bunch of microservices

Testing a Microservice



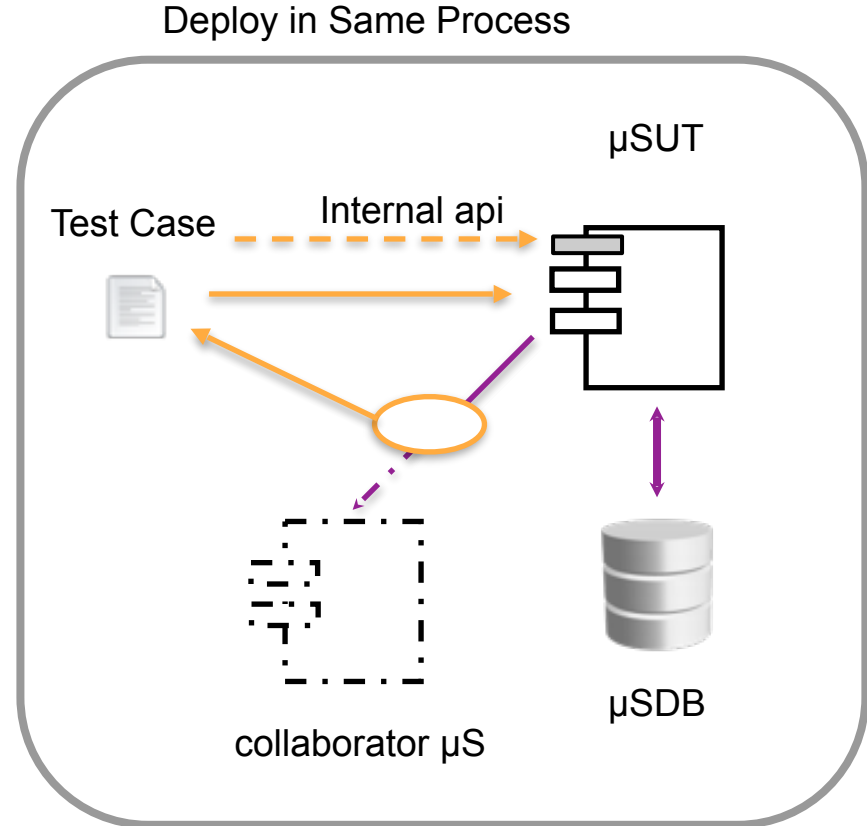
Microservice Tests

- Deploy your service on localhost (and the database if it has one)
- Test uses the public API to trigger functionality
- Mock a response to any calls made to other services or queues



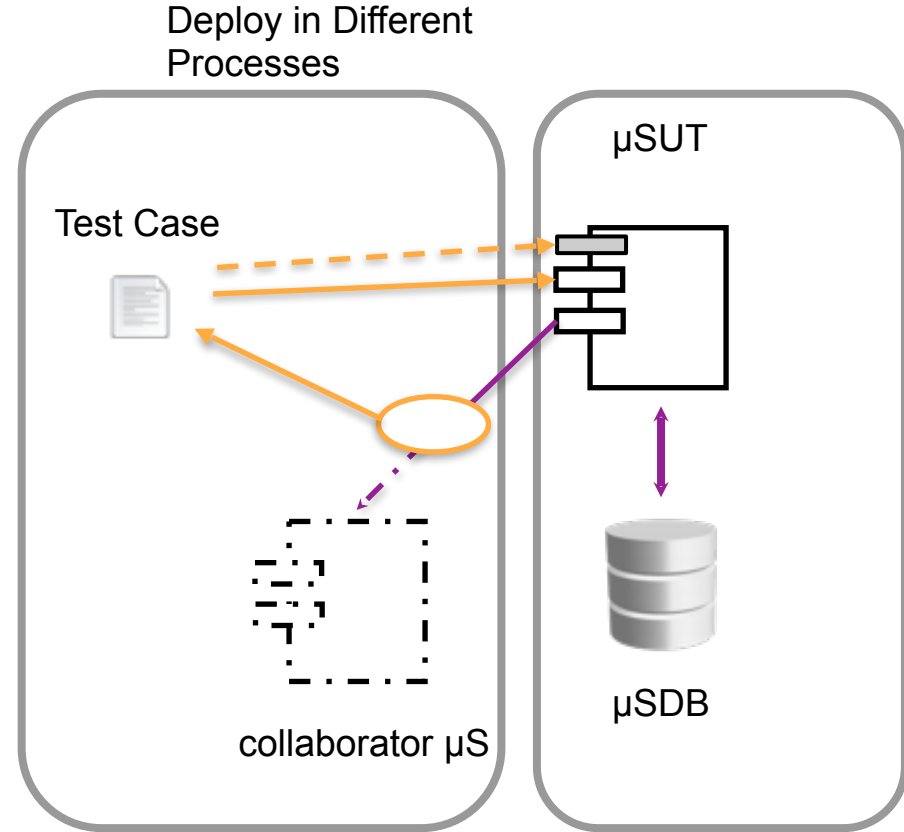
Faster execution

- In-process testing: Deploy your service with an in-memory database and in-memory api
- 'internal api' to set up test data and query internals



More production-like

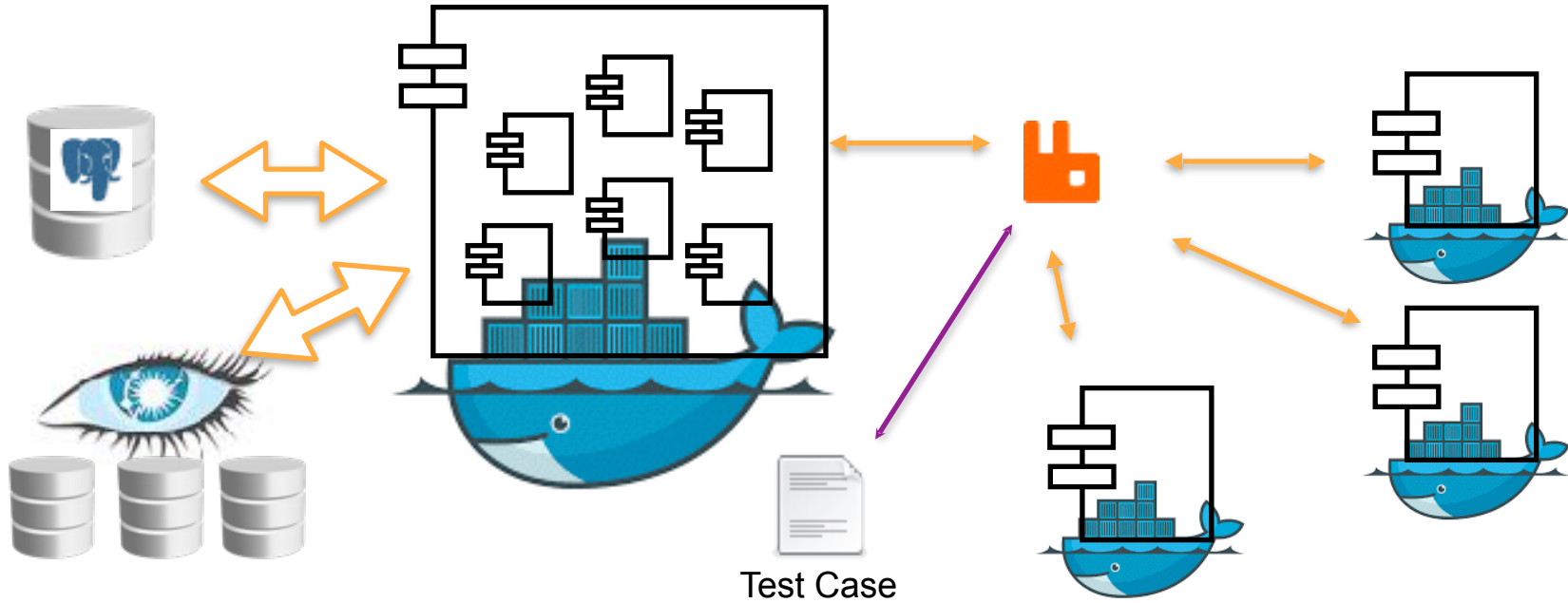
- Out-of-process testing: Deploy your service in one process, and the test case in another
- Security needed for 'internal api' so only test/non-production code can access
- Easier to performance test



Business-Facing Feature Tests

Test a whole scenario across several microservices

Testing a whole Feature



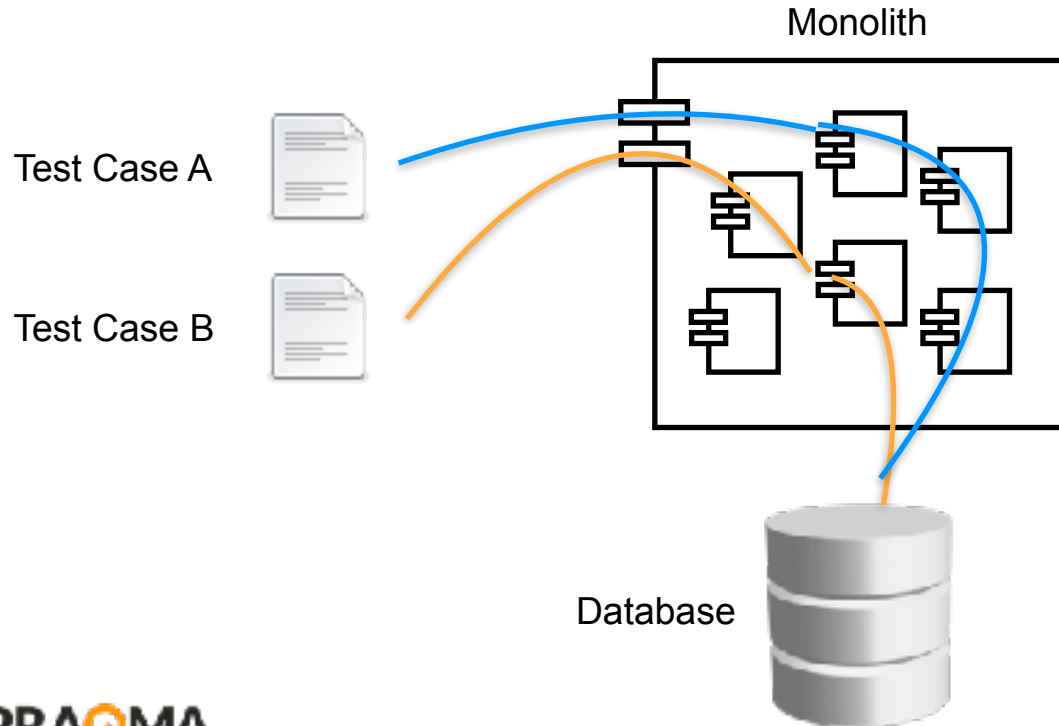
This is where it starts getting really complicated...

Example End-to-End Test Strategy

- Test crucial workflows, not every detail
- Use APIs instead of the GUI for most tests
- Techniques:
 - *Selective Deployment*
 - *Approval Testing*
 - *Event Monitoring*

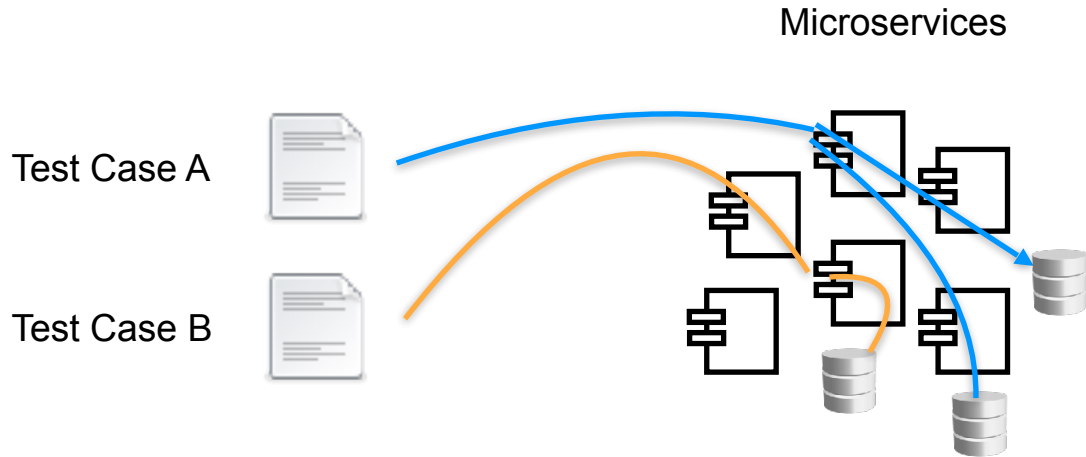
Keep testing costs
as low as possible

Service-layer tests in a monolith



- Test cases are for different business-facing workflows
- Will exercise different components within the monolith

Feature tests in microservices



- Test cases are for different business-facing workflows
- Will exercise different groups of microservices
- Need not deploy everything for every test case

Selective Deployment



- Need not deploy everything for every test case
- But: all test cases needed a few essential infrastructure services.

Pagero Online
Cloud Service



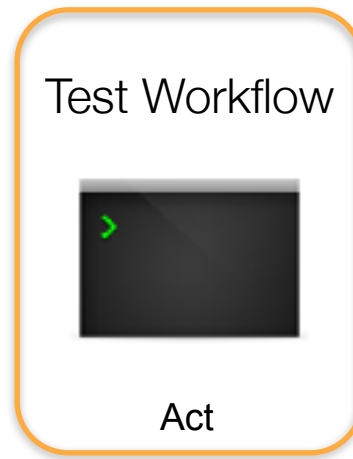
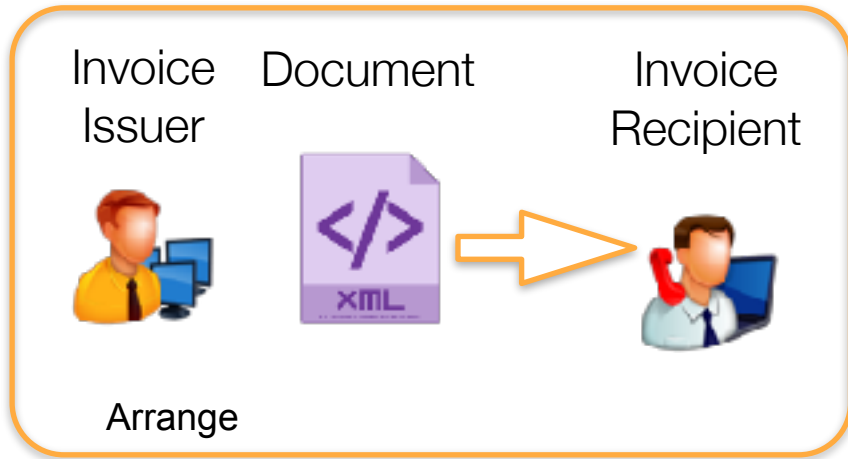
Invoice
Issuer



Invoice
Recipient

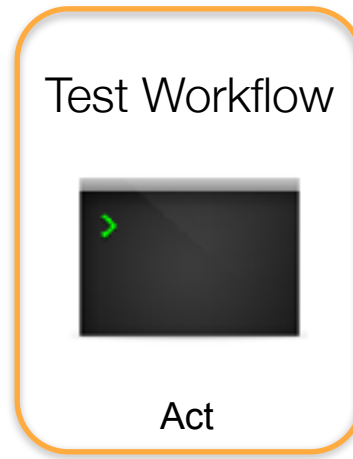
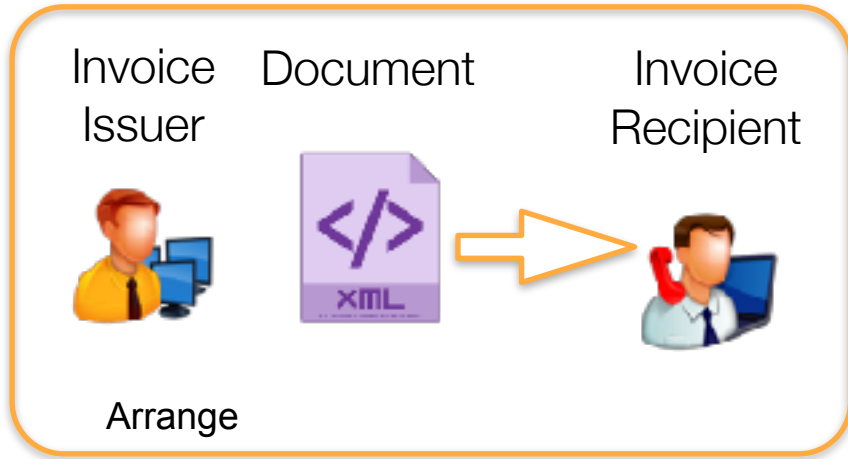


Parts of a test case



(Almost) all the tests involve these elements

Approval Testing



Approval Testing specifies the Assert: compare against an 'Approved' result



The “Approved” Result as Text

Recipient
Presentation



pdf-to-text utility

Recipient
Presentation
in plain text



- Find the important outputs. Convert them to plain text.
- Use textual diff to decide if the actual output matches the approved version

Elements of a Test Case

Invoice
Issuer

Document

Invoice
Recipient



Input data varies by test case

Test Workflow



common to
many tests

Recipient
Presentation



for reference

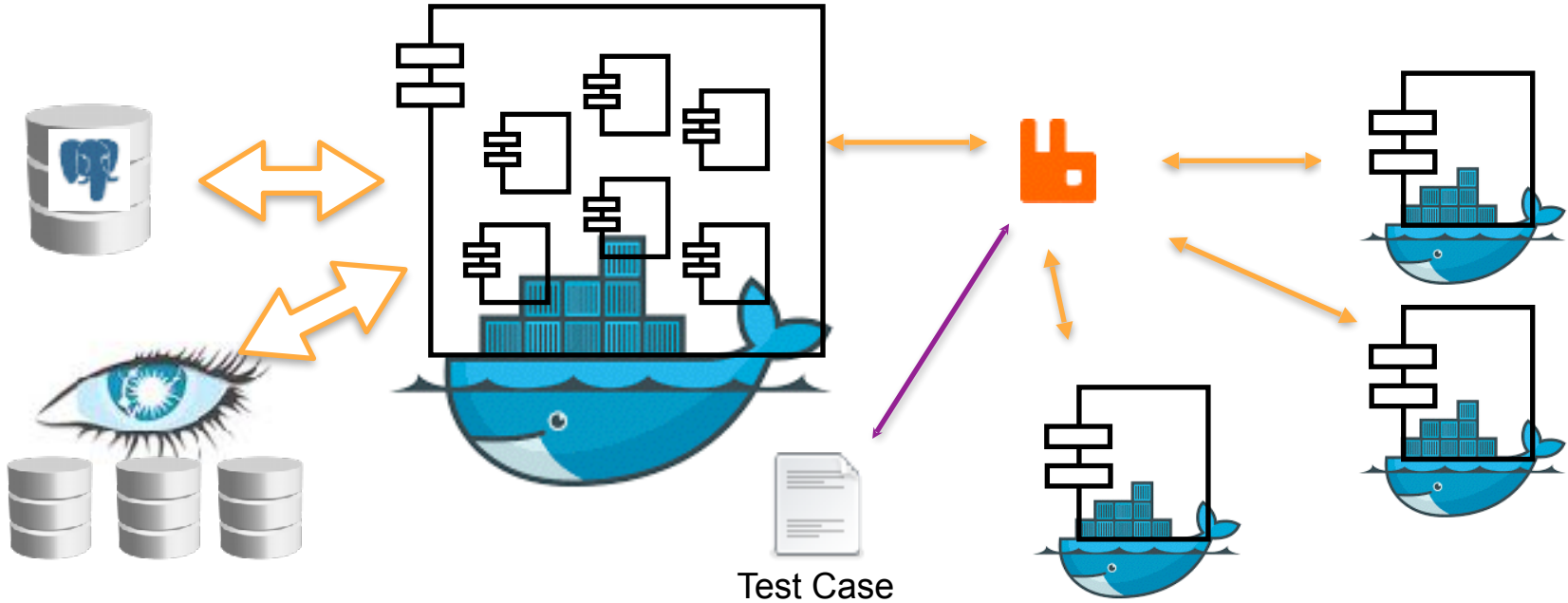
Approved
Output



to determine
pass/fail

- Minimise test creation work
- Minimise test maintenance work
- Maximise serendipity - find bugs you didn't anticipate

Debugging a failing test?



Which part broke?

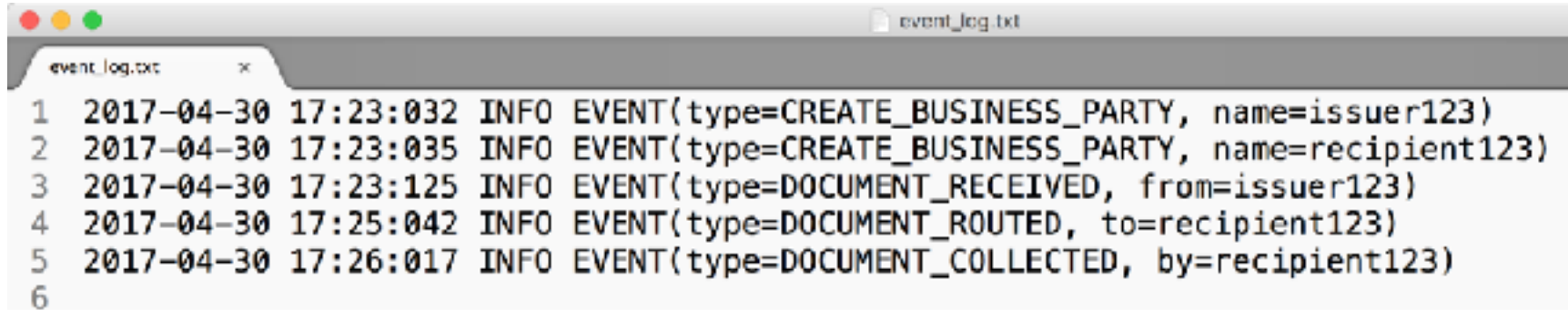
Correlation ID



Pick out all events with '123'

```
event_log.txt
2017-04-30 17:23:032 INFO EVENT(type=CREATE_BUSINESS_PARTY, name=issuer123)
2017-04-30 17:23:035 INFO EVENT(type=CREATE_BUSINESS_PARTY, name=recipient123)
2017-04-30 17:23:125 INFO EVENT(type=DOCUMENT_RECEIVED, from=issuer123)
2017-04-30 17:24:033 INFO EVENT(type=CREATE_BUSINESS_PARTY, name=issuer456)
2017-04-30 17:25:442 INFO EVENT(type=DOCUMENT_ROUTED, to=recipient123)
2017-04-30 17:26:016 INFO EVENT(type=DOCUMENT_COLLECTED, by=recipient123)
2017-04-30 17:27:022 INFO EVENT(type=CREATE_BUSINESS_PARTY, name=recipient890)
2017-04-30 17:28:057 INFO EVENT(type=DOCUMENT_COLLECTED, by=recipient456)
2017-04-30 17:29:742 INFO EVENT(type=DOCUMENT_ROUTED, to=recipient890)
```

Event log recorded in a test case

A screenshot of a text editor window titled 'event_log.txt'. The window contains a list of six log entries, each starting with a line number (1-6), a date and time stamp, a log level (INFO), and a detailed event description. The events describe the creation of business parties and the flow of a document through various stages: received, routed, and collected.

```
1 2017-04-30 17:23:032 INFO EVENT(type=CREATE_BUSINESS_PARTY, name=issuer123)
2 2017-04-30 17:23:035 INFO EVENT(type=CREATE_BUSINESS_PARTY, name=recipient123)
3 2017-04-30 17:23:125 INFO EVENT(type=DOCUMENT_RECEIVED, from=issuer123)
4 2017-04-30 17:25:042 INFO EVENT(type=DOCUMENT_ROUTED, to=recipient123)
5 2017-04-30 17:26:017 INFO EVENT(type=DOCUMENT_COLLECTED, by=recipient123)
6
```

Helps you to debug what happened when the test fails

Test Case Elements

Invoice
Issuer



Document



Invoice
Recipient



Input data varies by test case

Test Workflow



common to
many tests

Event Logs



to debug
the test

Approved
Output



to determine
pass/fail

- Recorded traffic is stored with the approved output

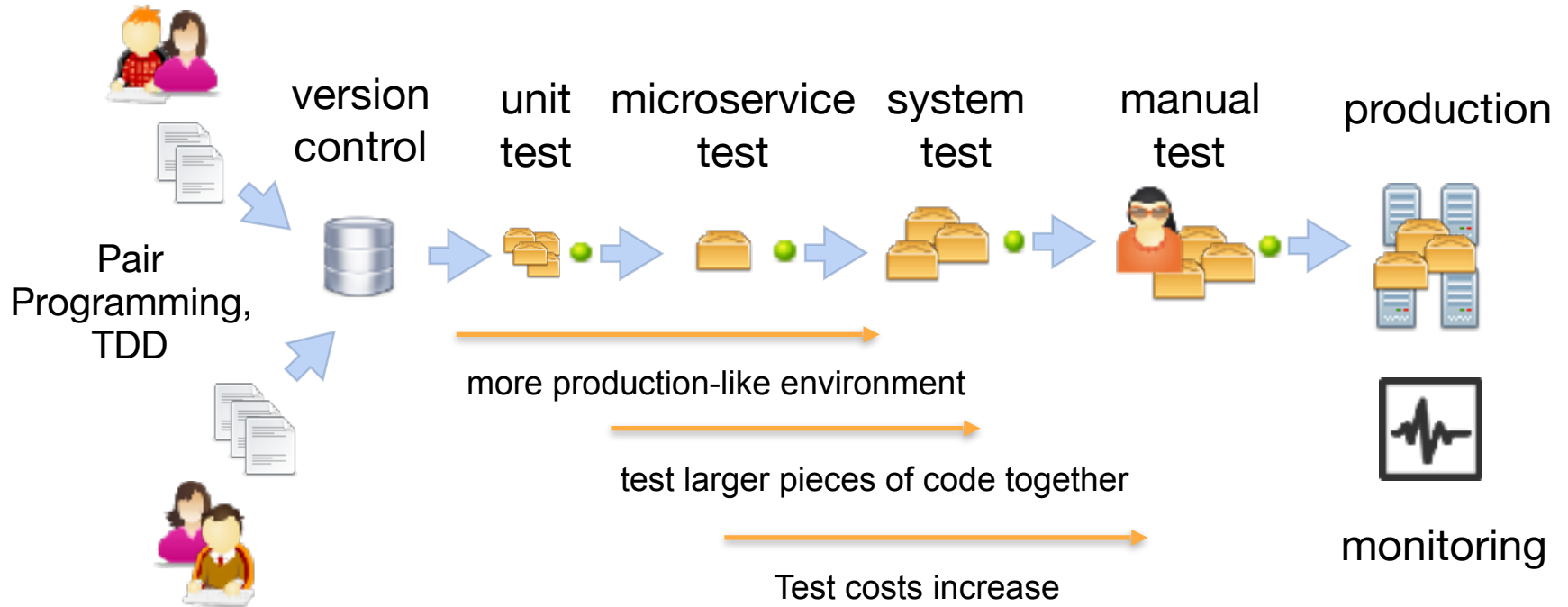
Recipient
Presentation
for reference



Independent Microservices & Teams

Organizing Testing efforts

Testing in the pipeline



Multi-team development



Component A



Infrastructure & Architecture

Component B

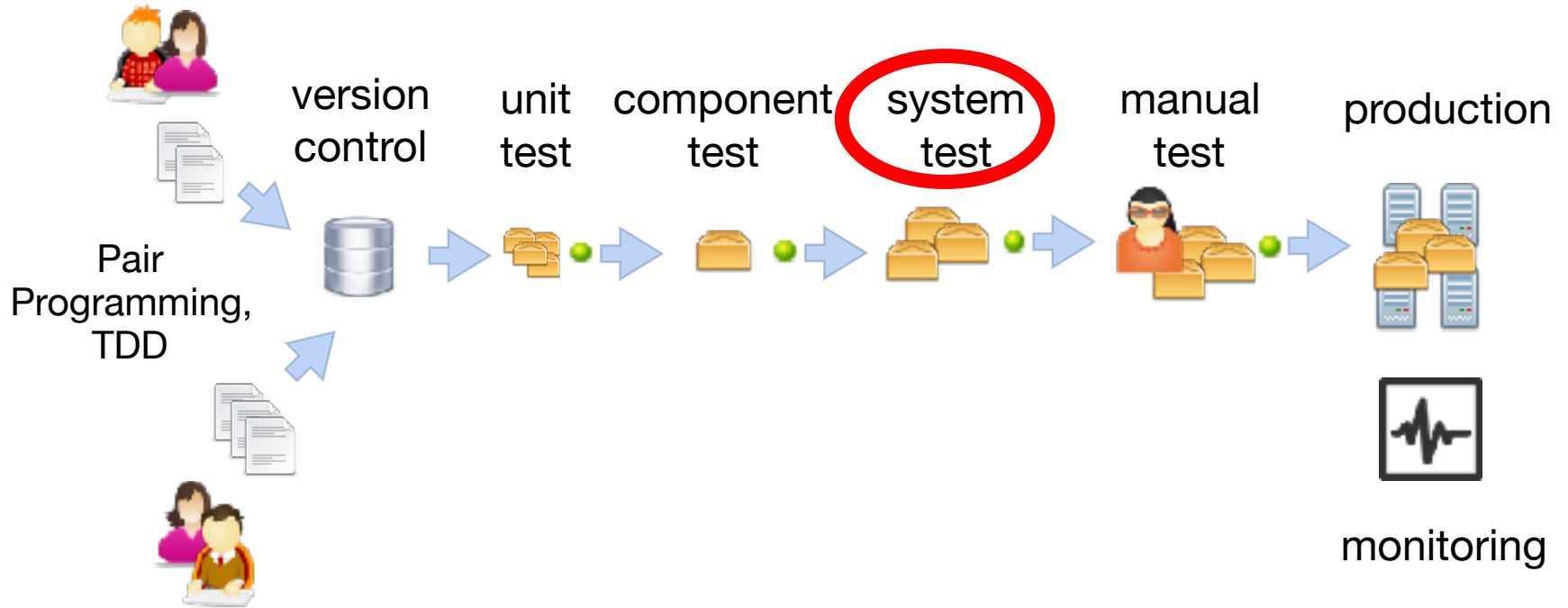


Components C&D

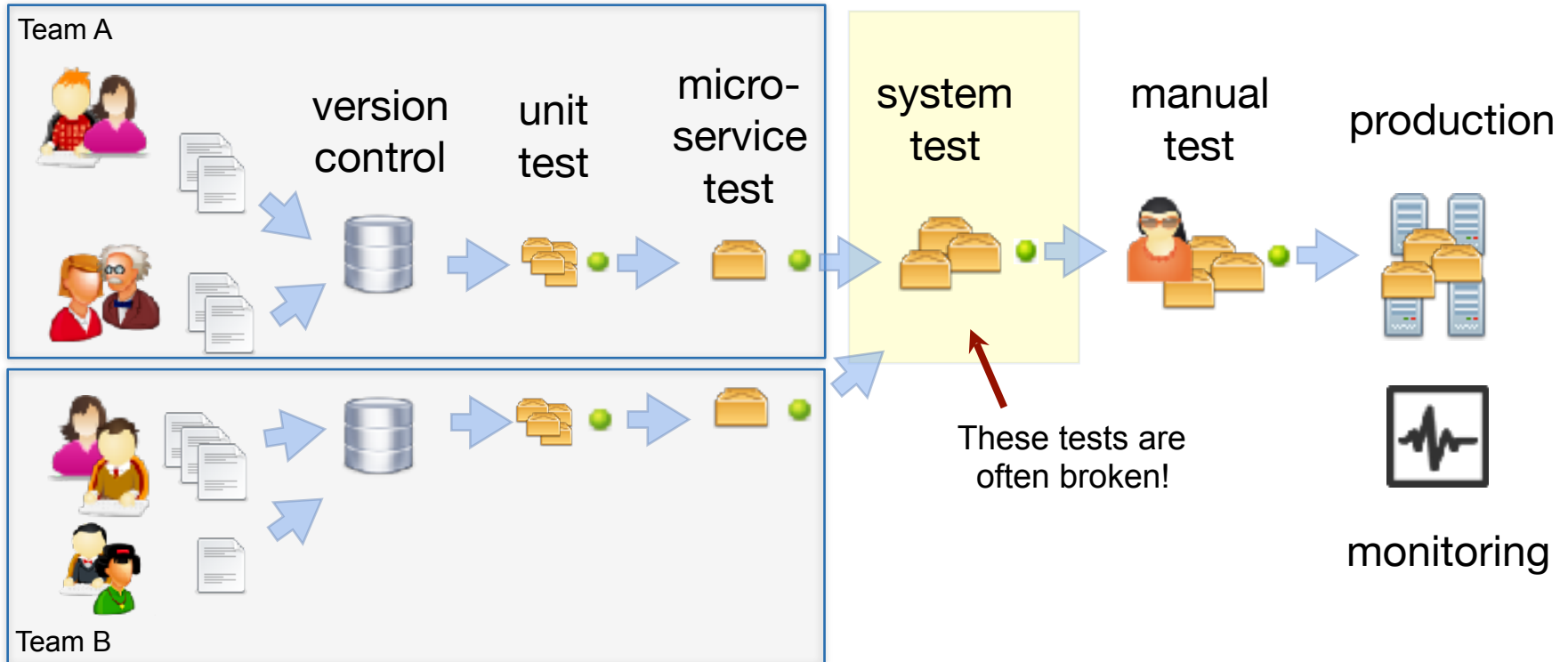


Components E&F

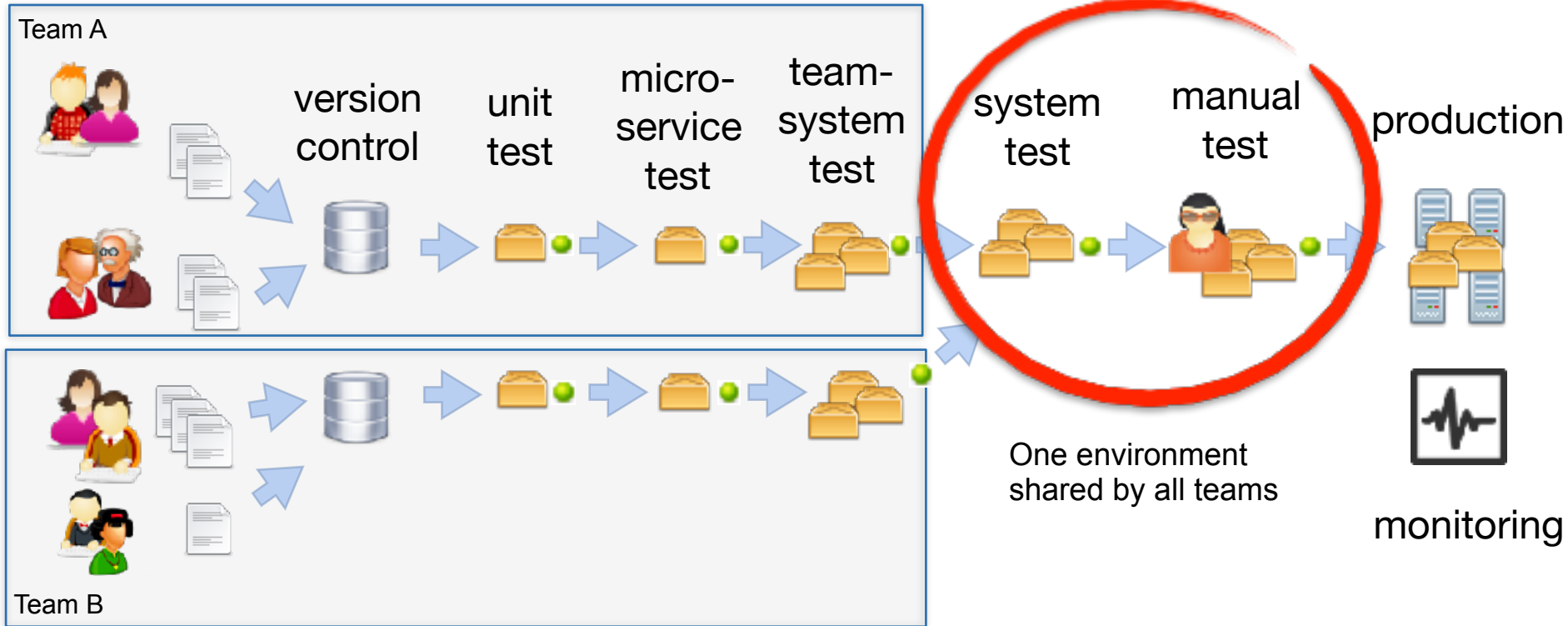
Testing in the pipeline



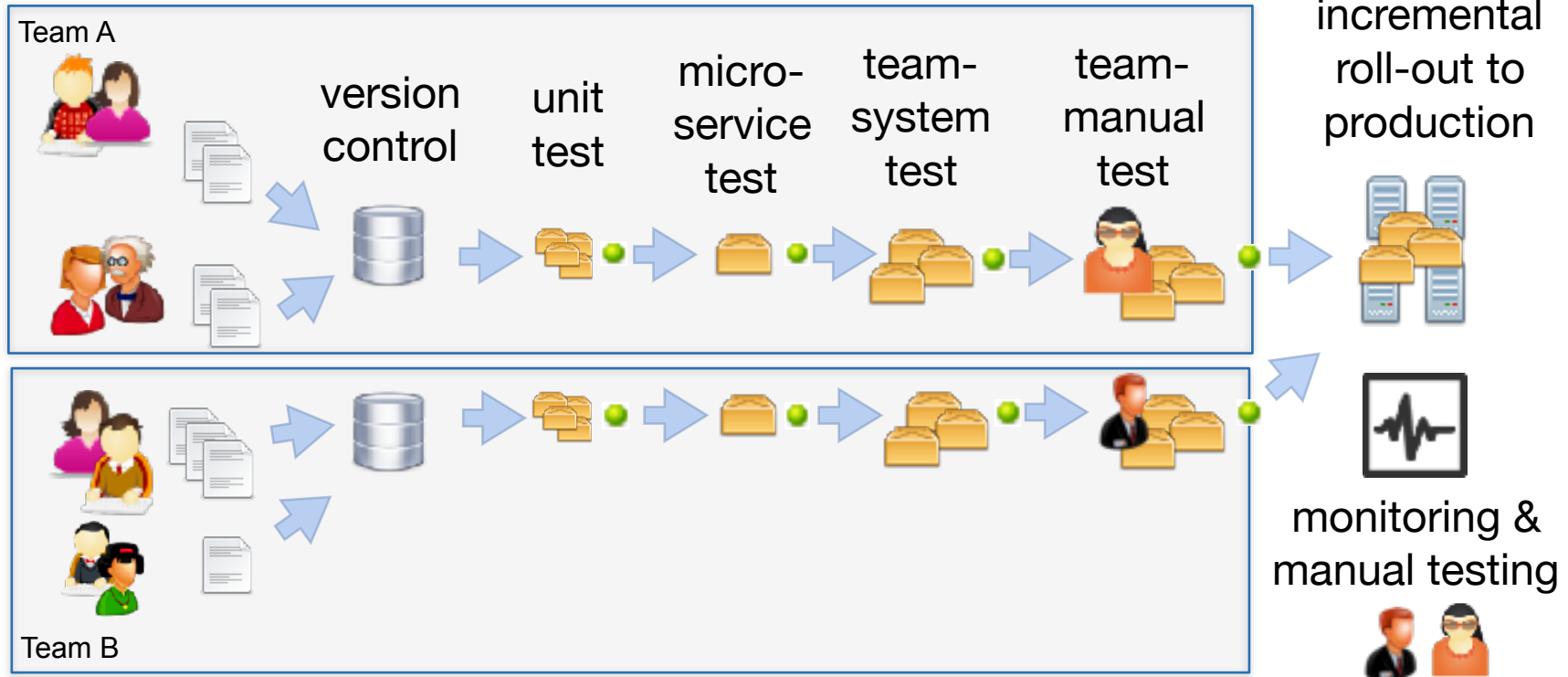
Multi-Team CD



Team pipelines



Teams Deploy Independently





Pair Programming, TDD

version control



unit test



micro-service test



system test



manual test



production



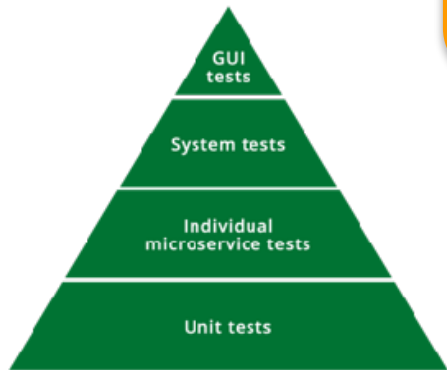
monitoring



@emilybache

 TextTest

Testing in a Microservices Architecture



team-system test

System Tests



PRAQMA



Pair Programming, TDD

version control



unit test



micro-service test



system test



manual test



production



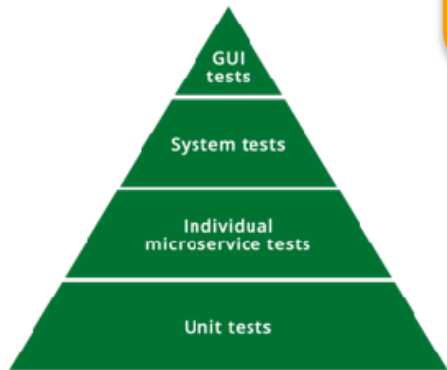
monitoring



@emilybache

 TextTest

Testing in a Microservices Architecture



PRAQMA

System Tests



team-system test



PRAQMA